AMENDMENTS TO THE CLAIMS

Please amend claims 1 and 15 without acquiescence to the basis of the rejections set forth in the Office Action, and without prejudice to pursue the original claims or previously presented claims in a related application. A complete listing of the pending claims is provided below.

1. (Currently Amended) A computer-implemented method for processing a program statement in a database query language, the program statement corresponding to a plurality of operators, wherein an operator tree is associated with the plurality of operators, the operator tree comprising a parent operator node, the method comprising:

identifying a child node that is associated with the parent operator node;

determining if the child node relates to an operator for which top-down processing is capable of being performed, wherein the top-down processing is capable of being performed when a result for the operator is capable of being generated without storage of the result for the parent operator node;

calling and executing the operator for the child node to generate a result using a processor; and

outputting the result to a data stream without buffering the result or an intermediate result in storage when top-down processing is performed.

2. (Original) The method of claim 1 further comprising:

determining whether the data stream already exists; and

creating the data stream if it does not exist.

3. (Original) The method of claim 1 in which the program statement is intended to create XML, wherein one or more XML tags are generated.

4. (Original) The method of claim 3 in which the program statement comprises a SQL/XML operator.

5. (Original) The method of claim 4 in which the SQL/XML operator is a XMLElement(), XMLAgg(), XMLConcat(), XMLForest(), XMLAttribute(), XMLComment(), or XMLPI() operator.

6. (Original) The method of claim 1 in which nodes corresponding to a concatenate operation or a CASE WHEN statement on top of SQL/XML operator are eligible for top-down processing.

7. (Original) The method of claim 1 in which the data stream is closed after the parent operator node has been fully evaluated.

8. (Previously Presented) The method of claim 1 further comprising identifying another child operator node, wherein the another child operator node is not eligible for top-down processing.

9. (Previously Presented) The method of claim 8 in which the another child operator node is evaluated using bottom-up processing.

10. (Original) The method of claim 8 in which both top-down and bottom-up processing are used to evaluate the program statement.

11. (Previously Presented) The method of claim 1 in which the data stream is built at an intended target location.

12. (Original) The method of claim 1 in which the data stream is a single data stream.

13. (Original) The method of claim 1 in which the data stream is built on a buffer, LOB, HTTP stream, segmented array, data socket, pipe, file, internet stream type, network stream type, or FTP stream.

14. (Previously Presented) The method of claim 1 in which an intermediate copy is not stored for the output result.

15. (Currently Amended) A computer-implemented method for processing a program statement, the program statement corresponding to a plurality of operators, wherein an operator tree can is associated with the plurality of operators, the operator tree comprising a parent operator node, the method comprising:

(a) determining whether the parent operator node is related to a first child operator node that is eligible for top-down processing, wherein the first child operator node is eligible for the top-down processing when a result for an operator associated with the first child operator node is capable of being generated without storage of the result for the parent operator node; and

(b) evaluating the first child operator node using a processor with top-down processing if the child operator is eligible for top-down processing, wherein the output from top-down processing the first child operator node is output to a data stream without buffering the result or an intermediate result in storage.

16. (Original) The method of claim 15 in which the program statement is intended to create XML, wherein one or more XML tags are generated.

17. (Original) The method of claim 16 in which the program statement comprises a SQL/XML operator.

18. (Original) The method of claim 17 in which the SQL/XML operator is a XMLElement(), XMLAgg(), XMLConcat(), XMLForest(), XMLAttribute(), XMLComment(), or XMLPI() operator.

19. (Original) The method of claim 15 in which nodes corresponding to a concatenate operation or a CASE WHEN statement over a SQL/XML operator are eligible for top-down processing.

20. (Original) The method of claim 15 in which an intermediate copy is not stored for the output from the first child operator node.

21. (Original) The method of claim 15 in which a second child operator node is identified which is not eligible for top-down processing.

22. (Previously Presented) The method of claim 21 in which the second child operator node not eligible for top-down processing is evaluated using bottom-up processing.

23. (Original) The method of claim 15 in which the data stream is built at an intended target location for the output from the first child operator node.

24. (Original) The method of claim 15 in which the data stream is a single data stream.

25. (Original) The method of claim 15 in which the data stream is built on a buffer, LOB, HTTP stream, segmented array, data socket, pipe, file, internet stream type, network stream type, or FTP stream.

26. (Previously Presented) A computer program product comprising a computer usable medium having executable code to execute a process for processing a program statement in a database query language, the computer usable medium comprising a volatile or non-volatile medium, the program statement corresponding to a plurality of operators, wherein an operator tree is associated with the plurality of operators, the operator tree comprising a parent operator node, the process comprising:

identifying a child node that is associated with the parent operator node;

determining if the child node relates to an operator for which top-down processing is capable of being performed, wherein the top-down processing is capable of being performed

when a result for the operator is capable of being generated without storage of the result for the parent operator node;

calling and executing the operator for the child node to generate a result; and

outputting the result to a data stream without buffering the result or an intermediate result in storage.

27. (Previously Presented) A system for processing a program statement in a database query language, the program statement corresponding to a plurality of operators, wherein an operator tree is associated with the plurality of operators, the operator tree comprising a parent operator node, the system comprising:

means for identifying a child node that is associated with the parent node;

means for determining if the child node relates to an operator for which top-down processing is capable of being performed, wherein the top-down processing is capable of being performed when a result for the operator is capable of being generated without storage of the result for the parent operator node;

means for calling and executing the operator for the child to generate a result; and

means for outputting the result to a data stream without buffering the result or an intermediate result in storage.

28. (Previously Presented) A computer program product comprising a computer usable medium having executable code to execute a process for processing a program statement, the computer usable medium comprising a volatile or non-volatile medium, the program statement

corresponding to a plurality of operators, wherein an operator tree is associated with the plurality of operators, the operator tree comprising a parent operator node, the process comprising:

(a) determining whether the parent operator node is related to a first child operator node that is eligible for top-down processing, wherein the first child operator node is eligible for the top-down processing when a result for an operator associated with the first child operator node is capable of being generated without storage of the result for the parent operator node; and

(b) evaluating the first child operator node with top-down processing if the child operator is eligible for top-down processing, wherein the result from the first child operator node is output to a data stream without buffering the result or an intermediate result in storage.

29. (Previously Presented) A system for processing a program statement, the program statement corresponding to a plurality of operators, wherein an operator tree is associated with the plurality of operators, the operator tree comprising a parent operator node, the system comprising:

(a) means for determining whether the parent operator node is related to a first child operator node that is eligible for top-down processing, wherein the first child operator node is eligible for the top-down processing when a result for an operator associated with the first child operator node is capable of being generated without storage of the result for the parent operator node; and

(b) means for evaluating the first child operator node with top-down processing if the child operator is eligible for top-down processing, wherein the result from the first child operator node is output to a data stream without buffering the result or an intermediate result in storage.

30. (Previously Presented) The computer program product of claim 26, wherein the program statement is intended to create XML, wherein one or more XML tags are generated.

31. (Previously Presented) The computer program product of claim 30, wherein the program statement comprises a SQL/XML operator.

32. (Previously Presented) The computer program product of claim 31, wherein the SQL/XML operator is a XMLElement(), XMLAgg(), XMLConcat(), XMLForest(), XMLAttribute(), XMLComment(), or XMLPI() operator.

33. (Previously Presented) The system of claim 27, wherein the program statement is intended to create XML, wherein one or more XML tags are generated.

34. (Previously Presented) The system of claim 33, wherein the program statement comprises a SQL/XML operator.

35. (Previously Presented) The system of claim 34, wherein the SQL/XML operator is a XMLElement(), XMLAgg(), XMLConcat(), XMLForest(), XMLAttribute(), XMLComment(), or XMLPI() operator.

36. (Previously Presented) The computer program product of claim 28, wherein the program statement is intended to create XML, wherein one or more XML tags are generated.

37. (Previously Presented) The computer program product of claim 36, wherein the program statement comprises a SQL/XML operator.

38. (Previously Presented) The computer program product of claim 37, wherein the SQL/XML operator is a XMLElement(), XMLAgg(), XMLConcat(), XMLForest(), XMLAttribute(), XMLComment(), or XMLPI() operator.

39. (Previously Presented) The system of claim 29, wherein the program statement is intended to create XML, wherein one or more XML tags are generated.

40. (Previously Presented) The system of claim 39, wherein the program statement comprises a SQL/XML operator.

41. (Previously Presented) The system of claim 40, wherein the SQL/XML operator is a XMLElement(), XMLAgg(), XMLConcat(), XMLForest(), XMLAttribute(), XMLComment(), or XMLPI() operator.

42. (Previously Presented) The system of claim 27, further comprising a volatile or non-volatile medium for storing information regarding the child node.

43. (Previously Presented) The system of claim 29, further comprising a volatile or non-volatile medium for storing information regarding the first child node.